Background

- Memory protections are commonly deployed today
  - largely prevents code injection
- Focus has shifted to code reuse attacks
  - e.g. ROP, JOP
- Various mitigations today
  - e.g. ASLR, execute-only memory, CFI, canaries, pointer mangling, shadow stacks
  - not as widely deployed
  - can be difficult to integrate
  - can have non-trivial performance / code size impact
  - can inhibit debugging
pointer authentication

- Optional ARMv8.3-A extension
- Detects illicit modification of pointers (and data structures)
  ... can be used to catch ROP, etc
  ... simple to integrate
  ... with minimal code size / performance impact
- Backwards compatible subset
  ... binaries using some features can run on any ARMv8-A CPU (without protection)
  ... so distributions only need one set of binaries
ROP protection example

```assembly
paciasp
stp fp, lr, [sp, #-FRAME_SIZE]!
mov fp, sp

< function body >

ldp fp, lr, [sp], #FRAME_SIZE
autiasp
ret
```

© ARM 2017
Pointer authentication basics

- New instructions to sign and authenticate pointers... against a user-chosen (dynamic) context
  ... e.g. return address is valid for a given stackframe
  ... architecture provides mechanism, not policy

- Uses a Pointer Authentication Code (PAC)
  ... authentication metadata stored within pointer
  ... so no additional space required
Each PAC is derived from:
- A pointer value
- A 64-bit context value
- A 128-bit secret key

PAC algorithm $P$ can be:
- QARMA\(^1\)
- IMPLEMENTATION DEFINED

Instructions hide the algorithm details

\(^1\)https://eprint.iacr.org/2016/444.pdf
Keys

- **Secret** 128-bit value
  - ... inhibit prediction / forging of PACs
- Held in system registers
  - ... can be used, but not read/written at EL0 (userspace)
  - ... limited risk of disclosure / modification
- Several keys:
  - APIAKey, APIBKey *(instruction pointers)*
  - APDAKey, APDBKey *(data pointers)*
  - APGAKey *(data)*
Pointers in AArch64

![Diagram showing pointer address fields in AArch64]

- Address
- Reserved
- Low/High
- Tag/Reserved
Pointers in AArch64 (with authentication)

- PAC embedded in reserved pointer bits
  - e.g. 7 bits with 48-bit VA with tagging
  - leaving remaining bits intact
Pointers in AArch64 (with authentication)

- PAC embedded in reserved pointer bits
  - e.g. 15 bits with 48-bit VA without tagging
  - leaving remaining bits intact
Operations: sign

- PAC* instructions sign pointers with PACs
- Result is not a usable pointer

Diagram:

- Pointer
- Context
- Key

Diagram nodes:

- P
- PAC
- +

Result:

- Pointer + PAC
Operations: authenticate

- **AUT* instructions authenticate PACs**
- If PAC matches, result is the original pointer
- If PAC doesn’t match, result is an invalid pointer → faults upon use
Operations: strip

- XPAC* instructions strip PACs
- Result is the original pointer
- No authentication is performed
Usage
ROP vulnerable code

```
stp fp, lr, [sp, #-FRAME_SIZE]!
mov fp, sp

< function body >

ldp fp, lr, [sp], #FRAME_SIZE
ret lr
```
ROP protection

pacia lr, sp
stp fp, lr, [sp, #-FRAME_SIZE]!
mov fp, sp

< function body >

ldp fp, lr, [sp], #FRAME_SIZE
autia lr, sp
ret lr
ROP protection (backwards compatible)

```assembly
paciasp
stp fp, lr, [sp, #-FRAME_SIZE]!
mov fp, sp

< function body >

ldp fp, lr, [sp], #FRAME_SIZE
autiasp
ret lr
```
Other uses

- Many potential uses / contexts:
  - locally-scoped pointers / stackframe
  - PLTs / PLT address (dynamic link time)
  - opaque pointers / logical type, owner

- Architecture provides mechanism, not policy
- needs careful consideration of reuse attacks
  - Need to avoid signing gadgets
  - May require multiple keys for distinct purposes
Software support
Linux Kernel

- RFCs\textsuperscript{2,3} posted
- Enables userspace use
  - ... per-process \texttt{APIAKey} initialized at exec() time
  - ... context-switched by kernel
  - ... retained across fork()
- Ptrace interface to find PAC bits (but not keys)
- Basic KVM support
- No kernelspace pointer authentication (yet)

\textsuperscript{2}https://lkml.kernel.org/r/1491232765-32501-1-git-send-email-mark.rutland@arm.com
\textsuperscript{3}https://lkml.kernel.org/r/1500480092-28480-1-git-send-email-mark.rutland@arm.com
Toolchain

- Upstream GCC 7 supports `-msign-return-address=[non-leaf | all]`
  ... uses APIAKey, backwards-compatible instructions (by default)
- GDB support pending kernel ptrace patches
- Thanks to Jiong Wang, Yao Qi
Questions?